Sprigenroller

Preparation

The random module from the Standard Python Library provides support for generating random numbers. The function random() from this module generates a random floating point number in the range $[0, 1[$. The function randint(a, b) can be used to generate a random integer from the range $[a, b]$. The following interactive Python session gives some examples on how to use these functions from the random module.

```
>>> import random
>>> help(random.random)
Help on built-in function random:

random(...)
    random() -> x in the interval [0, 1).
>>> random.random()
0.954131645221452
>>> random.random()
0.3548429482674793

>>> help(random.randint)
Help on method randint in module random:

randint(self, a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
>>> random.randint(3, 10)
5
>>> random.randint(3, 10)
8
```

Description

According to research at an English university, it doesn't matter in what order the letters in a word are. The only important thing is that the first and last letter are at the right place. The rest can be a total mess and you can still read it without a problem. This is because we do not read every letter by itself but the word as a whole.

If this is correct, it means the previous paragraph would still be perfectly readable if the words are distorted in the following way:

> *Aoccdrnig to a rscheearch at an Elingsh uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are. The olny iprmoetnt tihng is taht the frist and lsat ltteer is at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae we do not raed ervey lteter by itslef but the wrod as a wlohe.*

Assignment

1. Write a function distort_word that distorts a given word (passed as an argument to the function) according to the following rules:
    - the first letter of the distorted word is the same as the first letter of the given word

- the last letter of the distorted word is the same as the last letter of the given word
- the middle letters of the distorted word (all except the first and the last letter) are an anagram/permutation of the middle letters of the given word

The function must return the distorted word. To determine a random permutation p of a string s, the following procedure can be followed:

a. initialize the permutation p to the empty string
b. select a random letter from s
c. append the selected letter to p
d. remove the selected letter from s
e. repeat the previous procedure from step (b) until s contains no more letters

This results in the following sequence of steps that converts the word biologic into the random permutation ibiogolc, where we have underlined the selected letter of s in each step of the procedure:

s	$\rightarrow$	p
b<u>i</u>ologic		
<u>b</u>ologic		i
olog<u>i</u>c		ib
ol<u>o</u>gc		ibi
ol<u>g</u>c		ibio
<u>o</u>lc		ibiog
<u>l</u>c		ibiogo
<u>c</u>		ibiogol
		ibiogolc

2. Use the function distort_word to write a function distort_sentence, that distorts all words in a given sentence (passed as an argument to the function) according to the above procedure. The function must return the distorted sentence. The words of the sentence are defined as the longest possible sequence of letters. All characters that are not letters should remain unchanged and should retain their original position. As an example, if the string "Change, America needs change!" is passed to the function, the distorted string "Chnage, Ameicra nedes canhge!" could be returned. Note that this example contains two alternative distortions of the word change.

Example

```
>>> distort_word('monty')
'motny'

>>> distort_word('python')
'pothyn'

>>> distort_sentence('Nudge, nudge, wink, wink. Know what I mean?')
'Nugde, ngude, wnik, wnik. Konw what I maen?'

>>> distort_sentence('Nobody expects the Spanish inquisition!')
'Nbdooy epxtces the Snpasih iiiitqnsoun!'

>>> distort_sentence("He's not the Messiah - he's a very naughty boy.")
```

"He's not the Mseasih - he's a vrey nhaugty boy."

Voorbereiding

De module random uit de Standard Python Library kan onder andere gebruikt worden om willekeurige getallen te genereren. De functie random() uit deze module genereert een willekeurig *reëel* getal uit het interval $[0, 1[$. De functie randint(a, b) kan dan weer gebruikt worden om een willekeurig *geheel* getal te genereren uit het interval $[a, b]$. Onderstaande interactieve Python sessie geeft aan de hand van enkele voorbeelden aan hoe je de functies uit de module random kunt gebruiken.

```
>>> import random
>>> help(random.random)
Help on built-in function random:

random(...)
    random() -> x in the interval [0, 1).
>>> random.random()
0.954131645221452
>>> random.random()
0.3548429482674793

>>> help(random.randint)
Help on method randint in module random:

randint(self, a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
>>> random.randint(3, 10)
5
>>> random.randint(3, 10)
8
```

Omschrijving

Volgens onderzoek aan een Engelse universiteit speelt de volgorde van de letters in een woord geen rol, zolang de eerste en de laatste letter maar op de juiste plaats staan. De overige letters kunnen willekeurig door elkaar staan, zonder dat daarbij de leesbaarheid verloren gaat. Dit is een gevolg van het feit dat we niet elke letter afzonderlijk lezen, maar de woorden als een geheel.

Dit betekent dus dat de voorgaande paragraaf nog perfect leesbaar zou moeten zijn als de woorden op de volgende manier worden vervormd:

> *Vlogens oeoednrzk aan een Elgense usnrivteieit seeplt de vrdoogle van de letters in een woord geen rol, zlaong de erstee en de lttaase letter maar op de juiste platas saatn. De ovirege lertets knnuen wliueilekrg door ealkar staan zodner dat draibaj de lisaeraeehbd vleorern gaat. Dit is een govleg van het fiet dat we niet elke lteter aeordijzfnlk lzeen, maar de wroedon als een geeehl.*

Opgave

1. Schrijf een functie vervorm_woord die een gegeven woord (dat als argument aan de functie moet doorgegeven worden) vervormt, waarbij geldt dat:

- de eerste letter van het vervormde woord is gelijk aan de eerste letter van het originele woord
- de laatste letter van het vervormde woord is gelijk aan de laatste letter van het originele woord
- de middelste letters van het vervormde woord (alles behalve eerste en laatste letter) vormen een willekeurig anagram/permutatie van de middelste letters van het originele woord

De functie moet het vervormde woord als resultaat teruggeven. Om een willekeurige permutatie p van een tekenreeks s te bepalen moet je op de volgende manier te werk gaan:

a. de permutatie p is initieel gelijk aan de lege string
b. selecteer een willekeurige letter uit s
c. voeg de geselecteerde letter achteraan toe aan p
d. verwijder de geselecteerde letter uit s
e. herhaal voorgaande procedure vanaf stap (b) totdat s geen letters meer bevat

Dit levert bijvoorbeeld de volgende reeks van stappen op die het woord geologie omzet naar de willekeurige permutatie egiogole, waarbij de geselecteerde letter van s in elke stap onderlijnd wordt:

s	$\rightarrow$	p
geologie		
gologie		e
ologie		eg
ologe		egi
olge		egio
ole		egiog
le		egiogo
e		egiogol
		egiogole

2. Gebruik de functie vervorm_woord om een functie vervorm_zin te schrijven, die alle woorden uit een gegeven zin (die als argument aan de functie wordt doorgegeven) vervormt volgens de bovenstaande procedure. De functie moet de vervormde zin als resultaat teruggeven. De woorden uit de gegeven zin worden gevormd door de langst mogelijke opeenvolgingen van letters. Alle karakters die geen letter voorstellen moeten op hun oorspronkelijke plaats blijven staan in de vervormde zin. Zo moet de functie voor de gegeven zin "Change, america needs change!" de vervormde zin "Chnage, ameicra nedes canhge!" teruggeven.

Voorbeeld

```
>>> vervorm_woord('monty')
'motny'

>>> vervorm_woord('python')
'pothyn'

>>> vervorm_zin('Nudge, nudge, wink, wink. Know what I mean?')
'Nugde, ngude, wnik, wnik. Konw what I maen?'
```

```
>>> vervorm_zin('Nobody expects the Spanish inquisition!')
'Nbdooy epxtces the Snpasih iiiitqnsoun!'

>>> vervorm_zin("He's not the Messiah - he's a very naughty boy.")
"He's not the Mseasih - he's a vrey nhaugty boy."
```